REMARKS

This Application has been carefully reviewed in light of the Office Action mailed July 19, 2005. At the time of the Office Action, Claims 9-29 and 31-39 were pending in the Application. Claims 9-29 and 31-39 were rejected. Applicants cancel Claims 9-29 and 31-39 without prejudice or disclaimer, and add Claims 40-69. Applicants submit that these new claims add no new subject matter and are fully supported by the specification. Applicants respectfully request reconsideration and favorable action in this case.

Section 103(a) Rejections

Claims 9-16 were rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 5325531 issued to McKeeman et al. ("*McKeeman*") in view of U.S. Patent No. 5826077 issued to Blakeley et al. ("*Blakeley*"). Claims 17-29 and 31-39 were rejected under U.S.C. § 103(a) as being unpatentable over *McKeeman* in view of *Blakeley*, and further in view of U.S. Patent No. 5926819 issued to Doo et al ("*Doo*").

Although the cancellation of Claims 9-29 and 31-39 render the Examiner's rejections of Claims 9-29 and 31-39 moot, in the interest of expediting the prosecution of the present application, Applicants shall assume that the Examiner rejected Claims 40-69 based on *McKeeman*, *Blakeley*, and/or *Doo*.

In order to establish a *prima facie* case of obviousness of a claimed invention, all claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 U.S.P.Q. 580 (C.C.P.A. 1974). Furthermore, the teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art, not in Applicant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 U.S.P.Q.2d 1438 (Fed. Cir. 1991). Applicants respectfully submit that each and every element of Claims 40-69 are not found within the references cited by the Examiner and that there is no motivation to combine the references as suggested by the Examiner.

McKeeman generally discloses a rapid computer assisted software engineering and development system ("RCASE") that employs an "incremental module dependency analysis (make) facility," while *Blakeley* generally discloses an object-oriented query language that achieves a better integration between a query and an object-oriented host programming language. *McKeeman*, col. 3, ll. 33-64; *Blakeley*, col. 6, ll. 30-34. Applicants submit that, outside of Applicants' own disclosure, there is no motivation to combine the references as

suggested by the Examiner. This is compounded by the fact that *McKeeman* fails to teach, suggest, or disclose performing the dependency analysis on code objects stored in a database, while *Blakeley* fails to teach, suggest, or disclose using the object-oriented query language query a database as part of a dependency analysis. For at least this reason, the Examiner has failed to establish a motivation to combine the two references as suggested.

Even assuming there was a motivation to combine the references as suggested by the Examiner, Applicants submit that each and every element of Claims 40-69 are not found within the references cited by the Examiner.

Claim 40 recites:

> A method of generating dependency information for code objects stored in a database, comprising:
>
> recursively querying a database for one or more dependencies of procedural code objects stored in the database;
>
> receiving an indication from the database of one or more dependencies of procedural code objects stored in the database; and
>
> generating a dependency information tracking array based on the indication of one or more dependencies of procedural code objects.

Applicants respectfully submit that the *McKeeman-Blakeley-Doo* combination suggested by the Examiner fails to teach, suggest, or disclose each of these elements. For example, the *Blakeley* reference relied upon by the Examiner fails to teach, suggest, or disclose "recursively querying a database for one or more dependencies of procedural code objects stored in the database." Instead, *Blakeley* discloses an object-oriented query language apparatus and method in which hierarchies or graphs may be used to represent complex objects, which are recursively defined in terms of other objects. Col. 5, ll. 10-18. Allowing a complex object to be recursively defined, however, is not the same as "recursively querying a database for one or more dependencies of procedural code objects stored in the database" (emphasis added). In fact, nowhere in *Blakeley* is it disclosed that a database may be recursively queried at all. For at least this reason Claim 40 is allowable over the cited references.

The references cited by the Examiner also fail to teach, suggest, or disclose "generating a dependency information tracking array based on the indication of one or more dependencies of procedural code objects." Instead, the *McKeeman* reference relied upon by

the Examiner discloses using a software development tool to generate and store fine grain dependency graphs that identify dependencies between symbols within application modules of software applications. Col. 17, ll. 48-52. These fine grain dependency graphs, however, are not generated based on an indication received from a database as required by Claim 40. In fact, *McKeeman* includes no mention of a database and fails to disclose how these fine grain dependency graphs are even generated. For at least this reason, as well, Claim 40 is allowable over the cited references. As such, Applicants respectfully request the full allowance of Claim 40.

Claims 52, 60, 64, 65, and 69 each recite elements similar to those discussed above with regard to Claim 40. For example, Claims 52, 65, and 69 each recite "recursively querying a database for one or more dependencies of procedural code objects stored in the database" and "generating a dependency information tracking array based on the [indication] of one or more dependencies of procedural code objects." Claims 60 and 64 each recite a software module operable to "recursively query the database for one or more dependencies of procedural code objects stored in the database" and "generate a dependency information tracking array based on the [indication] of one or more dependencies of procedural code objects." Therefore, Applicants submit that Claims 52, 60, 64, 65, and 69 are allowable over the cited art, for example, for reasons similar to those discussed above with regard to Claim 40. Applicants respectfully request the full allowance of Claims 52, 60, 64, 65, and 69.

Claims 41-51, 53-59, 61-63, and 66-68 each depend, directly or indirectly, from Claims 40, 52, 60, 64, and 65. Therefore, Applicants submit that Claims 41-51, 53-59, 61-63, and 66-68 are also allowable over the cited art, for example, for reasons similar to those discussed above with regard to Claims 40, 52, 60, 64, and 65. As such, Applicants respectfully request the full allowance of Claims 41-51, 53-59, 61-63, and 66-68.

CONCLUSION

Applicants have made an earnest attempt to place this case in condition for allowance. For the foregoing reasons, and for other apparent reasons, Applicants respectfully request full allowance of all pending Claims. If the Examiner feels that a telephone conference would advance prosecution of this Application in any manner, the undersigned attorney for Applicants stands ready to conduct such a conference at the convenience of the Examiner.

Applicants believe no fee is due. However, should there be a fee discrepancy, the Commissioner is hereby authorized to charge any required fees or credit any overpayments to Deposit Account No. 02-0384 of Baker Botts L.L.P.

Respectfully submitted,

BAKER BOTTS L.L.P.
Attorneys for Applicants

Thomas J. Frame
Reg. No. 47,232
Phone: (214) 953-6675

Date: September 21, 2005

CORRESPONDENCE ADDRESS:

Customer Number: **05073**
Attorney Docket No.: 063170.6289